Australian Government
Department of Defence
Defence Science and
Technology Organisation

Anomaly Detection and Attribution Using Bayesian Networks

Andrew Kirk, Jonathan Legg and Edwin El-Mahassni

National Security and Intelligence, Surveillance & Reconnaissance Division

Defence Science and Technology Organisation

DSTO–TR–2975

ABSTRACT

We present a novel approach to anomaly detection in Bayesian networks, enabling both the detection and explanation of anomalous cases in a dataset. By exploiting the structure of a Bayesian network, our algorithm is able to efficiently search for local maxima of data conflict between closely related variables. Benchmark tests using data simulated from complex Bayesian networks show that our approach provides a significant improvement over techniques that search for anomalies using the entire network, rather than its subsets. We conclude with demonstrations of the unique explanatory power of our approach in determining the observation(s) responsible for an anomaly.

APPROVED FOR PUBLIC RELEASE

Published by

DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: 1300 333 362
Facsimile: (08) 7389 6567

AR No. 015-964
June, 2014

APPROVED FOR PUBLIC RELEASE

Anomaly Detection and Attribution Using Bayesian Networks

Executive Summary

Anomaly detection techniques allow us to identify and investigate cases in a dataset which are inconsistent with the remainder of that dataset. This is an important and valuable technique, allowing us to find incorrect sensor readings, or to detect suspicious activity in observations of ordinary behaviour. The anomaly detection techniques considered here are not rule based. They detect any case which does not follow the dominant patterns of behaviour observed in the dataset, rather than searching for a specific anomalous pattern. The advantage of this is the potential detection of a wide variety of anomalous behaviour. The downside is that detections require further analysis by human operators to explain the reason for the anomaly. We propose a novel approach to anomaly detection which exploits the inherent structure of Bayesian networks to consider only the observations which are actually responsible for an anomaly. This approach, which was tested against AIS data on Sydney Harbour, allows our algorithm to not only ignore non-anomalous information, but also to explain the cause of an anomaly to a human operator. Further, it has demonstrated the ability to pick up injected anomalies and provide an explanation to the operator.

Authors

Andrew Kirk
NSID

Andrew Kirk is an undergraduate student at the Australian National University in Canberra, where he is enrolled in a joint Bachelor of Science/Bachelor of Economics degree, majoring in Theoretical Physics. His research interests are in the application of statistical analysis in supporting decision making. Andrew completed this work while participating in DSTO's Summer Vacation Scholarship program in 2013-2014, under the supervision of Dr. Edwin El-Mahassni and Dr. Jonathan Legg.

Jonathan Legg
NSID

Jonathan Legg is with the Data and Information Fusion Group where he conducts research into machine learning for the fusion of data and higher level information for applications including anomaly detection.

Edwin El-Mahassni
NSID

Edwin El-Mahassni is with the Data and Information Fusion Group where he is currently interested in techniques for fusing uncertain and imprecise data.

Contents

1 Introduction **1**

1.1 Bayesian Networks for Anomaly Detection 1

2 Existing Methods of Bayesian Network Anomaly Detection **3**

2.1 Joint Probability 4

2.2 Jensen's Conflict Measure 4

3 The Conflict Localisation Algorithm **4**

3.1 The Starting Stage 5

3.2 The Optimisation Stage 6

3.3 The Explanation Stage 8

4 Possible Variations of the Conflict Localisation Algorithm **9**

4.1 Starting Rule 9

4.2 Traversal 11

5 Tests of Anomaly Detection **13**

5.1 Techniques for Evaluation of Classifier Performance 13

5.1.1 Receiver Operating Characteristic 13

5.1.2 Confusion Matrix 14

5.1.3 F_1 Score 14

5.1.4 Matthews Correlation Coefficient 14

5.1.5 Youden's J Statistic 15

5.2 Multiple Sources of Anomaly 15

5.3 Single Source of Anomaly 15

6 Tests of Anomaly Attribution **17**

6.1 Single Source of Anomaly 17

6.2 Multiple Sources of Anomaly 18

7 Conclusion **20**

References **21**

THIS PAGE IS INTENTIONALLY BLANK

1 Introduction

Underlying every issue in statistical reasoning is the assumption that the data being examined was generated by the same underlying process which our statistical models are designed to represent. The accuracy of a model compared to the process it represents is irrelevant unless we are considering data which was indeed generated by the same process. When this assumption fails for a given piece of data, that data is called an *outlier*, or an *anomaly*. Anomalies represent a crucial challenge in all areas of machine learning and statistics. Anomalies in a training set can compromise the validity of the resulting model and anomalies in data being fed into a forecasting model can lead to incorrect predictions. Anomalies are regularly a topic in their own right in the field of data mining, where we are interested in searching a dataset for anomalies for the purpose of finding data which appears to have been generated by a different process to the rest. This topic has attracted considerable attention in recent years, with anomaly detection being used to detect potential credit card fraud in financial transactions and disease outbreaks in medical data, to give some examples. A comprehensive review of existing anomaly detection techniques and applications can be found in [6].

The reasoning behind the power of anomaly detection in these cases is simple but powerful. It would be difficult if not impossible to build a model of how identity thieves use stolen credit cards, and searching for this behaviour directly would be futile. However, it is highly unlikely that they would behave identically to the true owner of the card. Using anomaly detection, we can determine which transactions are being generated by a different process (i.e. a different user), without having to specify what that process is. The techniques used for anomaly detection are many, varied, and complicated, and it is not our intent to detail them all here. As a brief treatment, most techniques consist of representing all observations in the Hilbert space of their variables, and using some distance or density measure to cluster them and return the smallest clusters as anomalies.

1.1 Bayesian Networks for Anomaly Detection

The techniques we are concerned with here all pertain to Bayesian networks (BNs). Introduced in [21] and codified in [22], BNs are a directed, acyclic graphical model, with evidence propagation governed by Bayes' theorem [16] (1). BNs are inherently robust to missing information, are better suited to categorical information than distance based classifiers [1], and their encoding of conditional and unconditional independence makes them highly efficient for calculating joint probability distribution functions in high-dimensional spaces [3]. Most importantly, BNs have a structure and a representation that is immediately understandable to human operators, even those who are not experts in machine learning, whereas many other machine learning techniques are nearly totally opaque [13, 14]. This structure allows us to model the flow of information through the network and thus trace the causes of anomalies, and output them in a way that makes sense to a human operator [2].

BNs represent each variable in a given dataset as a *node* in the network, denoted by a capital letter X. A node can take any state x in its state space $\mathcal{X}$. These state spaces are usually discrete and finite, though techniques exist for directly incorporating continuous

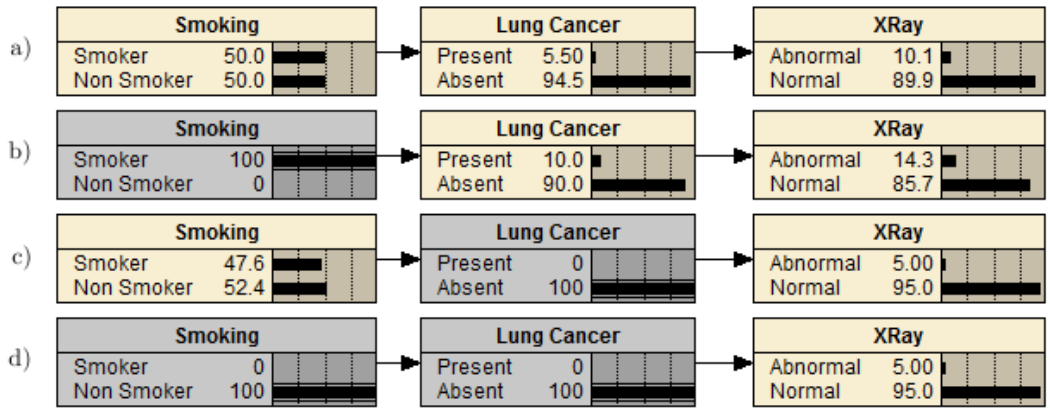

Figure 1: *Demonstration of causal independence in BNs*

variables in BNs, generally using Gaussian distributions [17] or using Gaussian mixtures to approximate other distributions [8]. Nodes are connected by causal links, known as *arcs* or *edges*, representing the causal dependence of one node on another. The notion of conditional independence is central to understanding BNs. Consider the BN shown in figure 1, adapted from the famous ASIA BN [12] and implemented in the *Netica 4.16* software package [7]. Figure 1b) shows that, absent any other knowledge, the fact that a person is a smoker increases our belief that they will have an abnormal X-Ray result. However, as figure 1c) and d) show, once we know that a person does not have lung cancer, our belief on them having an abnormal X-Ray result is unchanged by our knowledge of their smoking habits. We can that say that the nodes *XRay* and *Smoking* are conditionally independent given *Lung Cancer*, also written as $(XRay \perp Smoking)|Lung\ Cancer$.

In a BN, nodes with no parents (such as *Smoking* in figure 1) are given *prior probability distributions* over their states, based on our a priori beliefs about these nodes. A node X with parents $Y_1, Y_2, \ldots, Y_n$ has a *conditional probability distribution* $CPD : \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_n \mapsto \mathcal{X}$, which gives beliefs about node X given the beliefs about its parents. The distributions are governed by Bayes' Theorem:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}. \tag{1}$$

As a consequence of being inherently causal, BNs capture three distinct methods of information flow between nodes. Considering the case of figure 1c), we see that entering a *finding* on the node *Lung Cancer* unsurprisingly changes our beliefs about *XRay*. Here, by *finding*, we refer to having complete certainty regarding the state of a node. This causal flow from a parent to a child is known as *evidence propagation*. However, this finding also changes our beliefs in its parent, *Smoking*, as governed by (1). This reverse causal flow is called *inference*. Lastly, consider a variation of figure 1 which includes *Tuberculosis* as another parent of *XRay*. If we have a finding of "Abnormal" for the *XRay* node, then our belief in "Present" for *Tuberculosis* will increase via inference. However if we then add a finding of "Present" for *Lung Cancer*, ours beliefs for *Tuberculosis* will change again, as the finding on *Lung Cancer* means that we are no longer looking for something to explain

the finding of *X-Ray*. This is known as *explaining away*, and is of particular note because while it may seem intuitive to humans, most machine learning techniques are unable to perform this kind of reasoning.

Critical to the study of BNs are their junction tree decompositions [20]. Broadly speaking, the junction tree represents the propagation of information in a BN, in an undirected and noncyclical tree structure. This structure makes it easy to process, and junction trees lie at the heart of many engines for solving BNs computationally. The vertices of a junction tree are known as *cliques*, and each contain up to three nodes from the BN. Neighbouring cliques share at least one node, which serves as the connection through which information flows in the BN. Importantly, the fact that junction trees are undirected means that they are non-causal, and represent propagation and inference identically. An illustrative example of a junction tree is shown in figure 2.

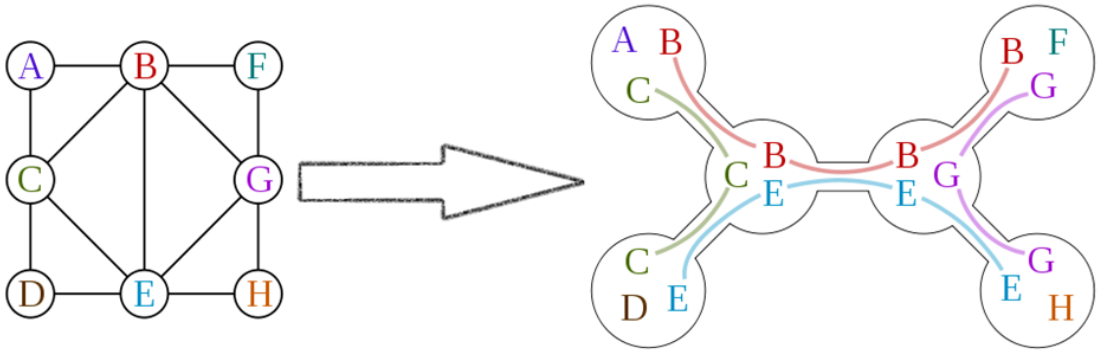

Figure 2: *An example of a junction tree decomposition*

2 Existing Methods of Bayesian Network Anomaly Detection

This report is not the first to discuss the topic of anomaly detection using BNs. However, most existing research focuses on techniques for learning the network from the data, rather than how to discriminate between anomalous and non-anomalous cases given the BN, and almost all existing research uses one of the two scoring methods detailed below. Both of these scoring methods classify anomalies based on the joint state of the entire network, rather than restricting their examination to the information relevant to the anomaly. The only example of a focused anomaly search we found in the existing literature was [2], which looks for anomalous results by searching for conflicts between parent-child pairs in the network. We note that this technique is severely limited in complex networks or networks where many nodes do not have findings. Moreover, this approach is completely unable to detect or explain more intricate anomalies arising from conflict between large sets of nodes, rather than individual nodes, or explain more intricate anomalies arising from conflicts between non-unit sets of data.

2.1 Joint Probability

Intuitively, the definition of an anomaly as an unexpected event is a sensible one. Although low-probability events can occur and be consistent with our model (else they would be zero-probability events), they certainly give us reason to be suspicious. As an extreme example, when in the midst of the 2007 US financial crisis, the Chief Financial Officer of Goldman Sachs claimed "We are seeing things that were 25-standard deviation moves, several days in a row,"[9] it was reasonable to conclude that the cause of the anomalies was a model that had lost touch with reality, rather than a severe case of bad luck.

We take the approach of [13], and use the information-theoretic measure of self-information $I(x) = -\log_2 P(x)$ rather than the joint probability itself. The transformation is strictly monotonic and thus has no effect on the actual classification of anomalies, but it provides a more easily human-readable score, and we will continue to refer to the metric as the joint probability.

2.2 Jensen's Conflict Measure

In order to differentiate between improbable, but internally consistent data and conflicting data, [10] introduced the conflict measure. Its use in the context of anomaly detection was discussed in [14]. For a set of observations $e = \{x_1, x_2, \ldots, x_n\}$, the conflict measure is defined as

$$\text{conf}(e) = \log_2 \frac{P(x_1)P(x_2)\cdots P(x_n)}{P(e)}, \tag{2}$$

where the terms in the numerator are the marginal priors of the model, before any evidence is entered. Importantly, the observations $x_1, \ldots, x_n$ do not necessarily indicate observations on a single node, but may themselves be sets of evidence similar to e. By including the naive probability for each observation or set thereof, as well as their joint probability, we are essentially testing against the expected correlation in our data, rather than simply its probability. The idea of using the conflict measure to trace sources of anomly is mentioned in [11], where the authors note that the fact that the conflict measure can be applied between sets of nodes rather than only individual nodes means that it can be used to search over partitions of the network, in order to locate the source of the conflict.

3 The Conflict Localisation Algorithm

Jensen's conflict provides a way to normalise the joint probability in subsets of the network and so compare conflict within these subsets. This, in turn, allows us to to find local maxima of conflict in the network, and isolate 'sources of conflict'. We expect these local maxima to provide a better means of anomaly detection by searching for anomalies directly, and not getting 'distracted' by parts of the network not immediately relevant to the anomaly at hand. Besides the pair-search technique in [2], who do not know of any existing methods with this property. Moreover, the detection of local maxima provides us with a means of expressing the nature of the anomaly to a human operator for further investigation. Existing classifiers can flag an observation as anomalous, but a report of

"the fifty variables in this case conflict with each other somehow" is far less valuable to a human than the output of our method, which provides a report of exactly where the conflict is located and which observations are most likely to be responsible.

We propose the Conflict Localisation Algorithm (CLA) as a computationally efficient method to both detect and explain anomalies in BNs. Inspired by the idea of 'tracing' conflict through the junction tree, detailed in [10, 11], the CLA is a multi-stage optimization algorithm, designed to exploit the existing structure of a Bayesian network in order to efficiently find local maxima of the Jensen conflict measure within the network. By finding local maxima, we not only overcome the issue of our anomaly classifier becoming 'distracted' by non-anomalous behaviour in other parts of the network, we are also able to directly report the source of the anomaly to a human operator. The algorithm works by first selecting cliques from the junction tree (CLA Part 1) to be passed to the second stage (CLA Part 2), where only the internal cliques above the mean are kept, and then it optimises these cliques (CLA Part 3) by finding the 'conflict set' of neighbouring nodes with the greatest conflict between them. CLA Part 1 is shown in algorithm 1.

Algorithm 1 The Conflict Localisation Algorithm, Part 1

```
1: procedure MAIN(junctionTree)
2:     internals ←FINDSTART(junctionTree)          ▷ This will be explained later
3:     branchedTo ← ∅
4:     topScore ← 0                                ▷ Largest optimised conflict
5:     topSet ← ∅                                  ▷ Conflict set with largest optimised conflict
6:     optScore ← ∅
7:     optSet ← ∅                                  ▷ Dictionary of form dict[key] = value
8:     for all clique ∈ internals do
9:         if clique ∉ branchedTo then
10:            optScore[clique], optSet[clique] ← OPTIMISE(clique,0,None)  ▷ This will be
   explained later
11:            branchedTo ← branchedTo ∪ optSet[clique]
12:            if optScore[clique] > topScore then
13:                topScore ← optScore[clique]
14:                topSet ← nodesIn(optSet[clique])  ▷ Save the nodes in the set of cliques
15:            end if
16:        end if
17:    end for
18:    return topScore, topSet
19: end procedure
```

3.1 The Starting Stage

In the first stage, the algorithm exhaustively searches over the junction tree corresponding to the network, calculating the Jensen conflict measure within each clique, as shown in algorithm 2. Note that because all nodes in a clique are directly connected, the flow of information and the calculation of the Jensen conflict measure is extremely simple within a clique, allowing us to search exhaustively without having to worry about the computational

burden. Since every edge in the network is contained within a clique we know that the 'frontier' between two conflicting sets (which may be single nodes, or larger collections of nodes) will exist within a clique, and thus the internal conflict of cliques should be a good way to find these conflicts. Multiple rules could be used for choosing which cliques to optimize over, we chose to find the mean of the internal conflict of cliques, and discard any clique with an internal conflict below the mean.

Algorithm 2 The Conflict Localisation Algorithm, Part 2

```
20: procedure FindStart(junctionTree)   ▷ Takes in junction tree with observations,
    returns cliques with internal conflict above the mean
21:     internals ← ∅                   ▷ Dictionary of form dict[key] = value
22:     for all clique ∈ junctionTree do
23:         internals[clique] ← conf(clique)
24:     end for
25:     avg ← mean(internals)
26:     for all scores ∈ internals do
27:         if score < avg then
28:             remove score
29:         end if
30:     end for
31:     internals ← order(internals)
32:     return internals
33: end procedure
```

3.2 The Optimisation Stage

The optimisation stage uses a recursive, hill climbing approach that searches for the 'conflict set' of nodes which give rise to local maxima of conflict within the junction tree. Although the result of this stage of the algorithm is a set of nodes, we are still using the junction tree structure, so this conflict set is built by searching over cliques, and when we say that 'a clique is added to the conflict set' we really mean that 'the nodes from the clique are added to the conflict set'. From a single staring clique, the optimiser tries to find a local maximum by searching over all neighbouring cliques in the junction tree, and checking if the confict measure of the observations contained within this new, larger conflict set is greater than the conflict measure of the observations in the previous conflict set. The neighbouring clique which gives the greatest increase in the conflict measure is then added to the set. This process continues as described in algorithm 3 until no neighbouring cliques increase the conflict measure, and the algorithm returns the current conflict set as a local maximum.

Searching over neighbouring cliques in the junction tree is not only computationally efficient (as it vastly narrows the search space, and the simple evidence propagation between neighbouring nodes makes computing the conflict measure easy), it has the important benefit of having an intuitive correspondence to the nature of conflicts. By definition, nodes which are related should be connected to each other in the network, and thus in its junction tree. The more distant a pair of nodes are, the more likely it is that a conflict between

Algorithm 3 The Conflict Localisation Algorithm, Part 3

```
34: procedure OPTIMISE(candidate, bestScore, confSetCliques)  ▷ Optimises from candidate clique, given existing conflict set. Returns the local maximum of conflict, and its associated conflict set.
35:     if bestScore = 0 then
36:         bestScore ← conf(candidate)
37:     end if
38:     score ← conf(candidate ∪ confSetCliques)
39:     if score < bestScore then
40:         return bestScore, confSetCliques  ▷ Return with no change if candidate does not add to conflict
41:     end if
42:     bestScore ← score  ▷ Update the maximum
43:     confSetCliques ← confSetCliques ∪ candidate
44:     branches ← connections(candidate) \ confSetCliques  ▷ Search space for next candidate
45:     while |branches| > 0 do
46:         bestBranchScore ← bestScore
47:         candidate ← None
48:         for all test ∈ branches do  ▷ Search over all branches first to pick the best
49:             branchScore ← conf(confSetCliques ∪ test)
50:             if branchScore > bestBranchScore then
51:                 bestBranchScore ← branchScore
52:                 candidate ← test
53:             end if
54:         end for
55:         if candidate = None then
56:             break
57:         end if
58:         bestScore, confSetCliques ← OPTIMISE(candidate,bestScore,confSetCliques)  ▷ Recursively call on best branch
59:         branches ← branches \ candidate  ▷ Remove used branch from search space
60:     end while  ▷ Return to search for second best branch
61:     return bestScore, confSetCliques
62: end procedure
```

them is a coincidence of correlation, rather than a disagreement that can be interpreted by a human. This comes with an important caveat however, as it relies on the 'true network' for an anomalous finding having a similar structure (though not probability distribution) as that of the non-anomalous findings used to build the network. As an example, in our tests below we used simulated data, where the findings for multiple nodes were switched with their findings from another case. This kind of simulation induces a relationship between these nodes which did not exist before, and is thus not captured by the network. Although this specific type of relationship is unique to simulated data, it is not unreasonable to anticipate real anomalies arising from similar database errors, or perhaps to have some actual relationship between nodes which only exists in anomalous cases. In these situations an anomaly will still be detected, but the CLA will not find the entire source.

In order to avoid redundancy, cliques that are included in any conflict set are removed from the list of starting cliques returned by the first stage of the algorithm. If the optimiser on clique α branches to clique β, representing a particular set of nodes in conflict, it makes sense that the same optimiser on clique β will simply branch to clique α and return the same result. Thus, by removing these cliques from the list of starting cliques we remove this redundancy, improving computational efficiency and simplifying output for human operators.

3.3 The Explanation Stage

A key advantage of the CLA over existing methods of anomaly detection is its ability to efficiently identify the cause of the anomalous classification, and to concisely report this information back to a human operator for further review. The CLA returns multiple conflicting sets for each case (one for each clique used to start the optimiser), although typically we are only interested in explanations for the few sets with the highest conflict. In order to find conflicts as efficiently as possible, the CLA searches for sets with large internal conflict within themselves. This means that the output of the CLA optimiser will contain anomalous and non-anomalous observations. The reason for this is that conflicts are between anomalous and non-anomalous data, not anomalous data alone. Whilst an algorithm which simply identifies all the relevant information and passes it to a human is certainly valuable, and a significant improvement over reporting the entire case, more can be done to improve this. The explanation stage of the CLA works in two substages, first identifying the internal 'tension' in the conflicting set reported by the optimiser, and then using the rest of the network in order to determine which part of the conflicting set is anomalous.

In order to identify the tension in the conflicting set we use another hill-climbing procedure. For each node in the conflict set $O \in \omega$, the CLA computes the conflict between the candidate node and remainder of the conflict set $\text{conf}(O, \omega \setminus O)$, in order to find the most conflicting node O^*. The conflict set is then broken down into two subsets $\alpha = O^*$ and $\beta = \omega \setminus O^*$. The CLA repeats this process recursively, searching for nodes in β to move to α in order to maximise $conf(\alpha, \beta)$. When no such nodes can be found, and we have a maximum of this conflict, the CLA returns α, β, and the conflict between them. We expect that one of these subsets will contain the anomalous observations in ω, whilst the other will contain the non-anomalous observations which the anomalous observations

are in conflict with. We note that it is certainly possible for multiple different sources of anomaly to exist near each other in the network, and that these may be better explained by more subsets, but we leave the best way to do this as a topic for future work.

A partitioned conflict set is valuable to a human operator, but an ideal explanation would identify which subset is the anomaly, and which is the evidence. Fortunately, we are able to do this by checking which subset is consistent with the remainder of the network η. We compute $\text{conf}(\alpha, \eta \setminus \omega)$ and $\text{conf}(\beta, \eta \setminus \omega)$, and make the determination that the subset with the greater conflict is the anomalous subset. Note that we explicitly do not consider β when computing the conflict for α, and vice versa. This is because we explicitly want to test against information that could not be part of the anomaly we are considering (else it would have been included in ω by the CLA). As an added measure, we also report the individual measures of consistency with the network for each node $\text{conf}(X, \eta \setminus (\omega \setminus X))$, which means X is in both measures. These scores have no effect on the actual processing of the CLA in explaining anomalies, but are a helpful measure for human operators, especially in the case of more complex anomalies.

4 Possible Variations of the Conflict Localisation Algorithm

The specification we have given for the CLA is not the only possible way the algorithm could be constructed, and it is not the only one we tried. It is however, in our experience, the best version of the algorithm for an arbitrary case, where we do not have prior information regarding the size of the dataset, the structure of the network, or the nature of the anomalies. We require that a 'good' construction of the CLA strike an appropriate balance between:

1. Computational efficiency
2. Performance in detecting anomalous cases
3. Performance in attribution of anomaly sources
4. Ability to detect and explain complex sources of conflict
5. Avoidance of redundancy in explanations

The tradeoff between these desires is determined by where we start the optimiser, and how it traverses the junction tree. Below, we discuss some alternative techniques for these to parameters, quantify their advantages and disadvantages, and explain why we chose the specification given above.

4.1 Starting Rule

Anomalies range from the simple case of a single node with an incorrect finding, to complex cases of multiple large sets of nodes which are internally consistent, but conflict with

each other. Nevertheless, all information in a BN must propagate along the edges of the network, and all anomalies of any nature will ultimately give rise to a conflict between two individual nodes, joined by an edge. As every edge is contained within some clique of the junction tree, we expect that every anomaly source will give rise to at least one clique with high internal conflict.

We note that it is certainly possible for many cliques with low internal conflict to return a conflict set with a large conflict when fed into the optimisation stage of the CLA, especially when we have complex anomaly sources. Indeed, figure 3, calculated from the datset described in section 5.2, shows this behaviour, and seems to support the notion that the relationship between internal and optimised conflict is extremely weak, if it exists at all. However, we are interested in finding the highest scoring anomaly sources, not in finding them as many times as possible, and not in finding every clique which could optimise to a given source. With that in mind, we look to see if our starting clique rule has any effect on the largest optimised conflict found for each case. Figure 4 compares the OPTIMISE output for the above average initialisation described in section 3.1 with the output using an exhaustive starting rule. The figure shows that the above average starting rule suffers a minimal loss of performance in this regard, with 60% of optimised conflicts being identical, and others being only slightly different. In our tests, we found that the exhaustive starting rule was over 67% slower on average when compared to the the above average starting rule.

Moreover, the above average starting rule had the qualitative benefit of producing more easily readable explanations of conflicts (item 5 in the above list), as it started at a location which was definitely part of the anomaly source. The exhaustive starting rule, however, produced several reports of the same anomaly source, sometimes starting from a non-anomalous clique which happened to be located next to the anomaly where it could

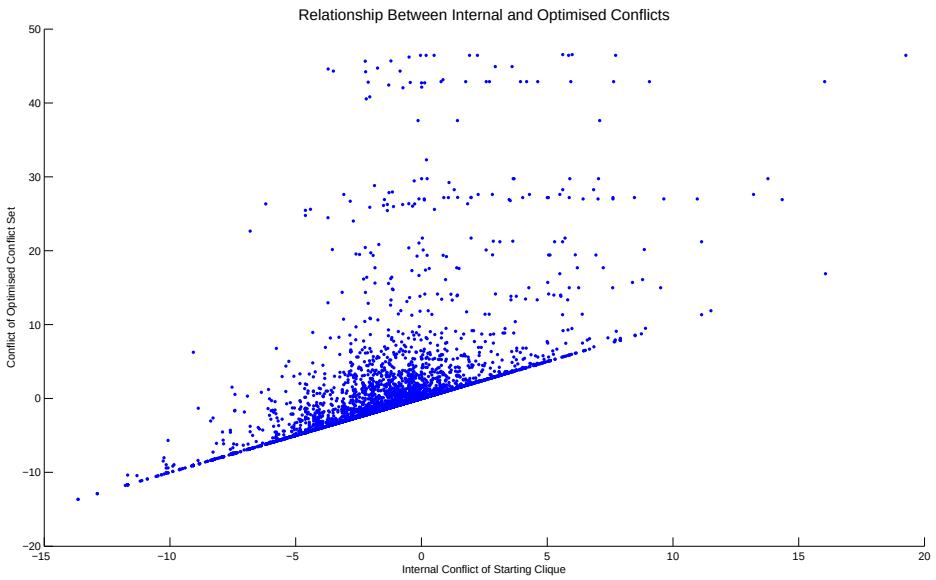

Figure 3: *Demonstration of low internal conflict cliques optimising to high conflict scores*

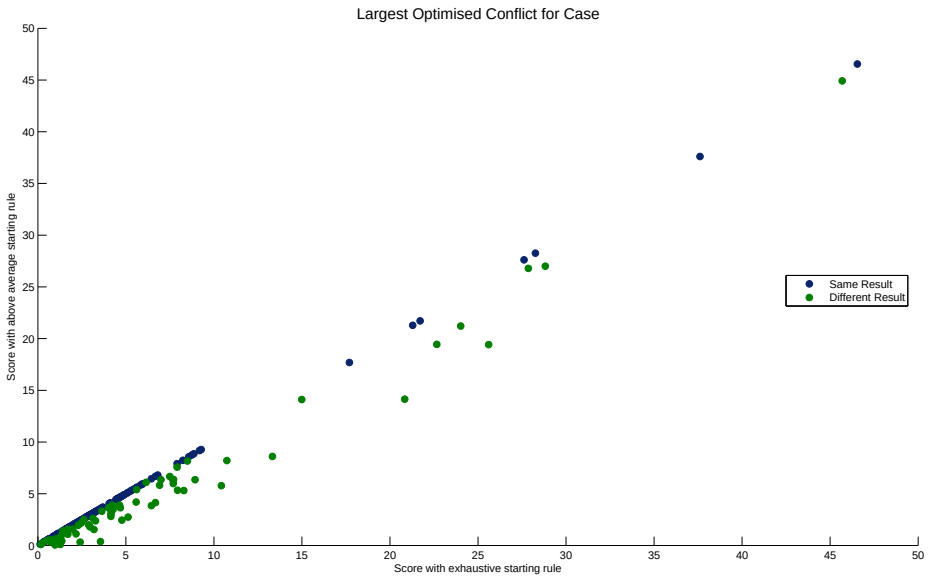

Figure 4: *Effect of starting clique rule on largest optimised conflict found*

optimise to it. The above average starting rule is certainly not the only possible alternative to the exhaustive rule, but as it gave a satisfactory improvement in computational efficiency and readability of explanations, without any noticeable decline in detection performance, we did not devote time to investigating other possible rules.

4.2 Traversal

The standard specification of the CLA given in algorithm 3 returns to previous cliques in order to search for second-best branches, which is certainly not the only possible technique. We also tested CLA without returning past cliques which only searched along the most promising branch (simply by eliminating the *for* loop at line 45 of the algorithm), a technique which we called *strict traversal*. Figure 5 shows the highest scoring optimised conflict for each case in the dataset from section 5.2, as found by strict traversal and by standard traversal. Whilst the standard traversal technique only found a different largest optimised conflict in 15% of cases, these results still give us considerable reason to prefer the standard traversal technique.

Firstly, there is considerable heteroskedasticity in these results. To be precise, it is in the cases with large optimised conflicts that we see large differences in the results from the two traversal techniques. This is a concern because it is these cases (the anomalous ones) that we are interested in. On the other hand, the two techniques tend to agree in the majority of low-conflict, non-anomalous cases, the ones we're not particularly interested in. Secondly, the (sometimes large) differences in conflict shown in figure 5 indicate that the standard traversal technique is finding additional cliques, particularly ones making large contributions to the optimised conflict, that the strict algorithm is not. This means that even when the strict traversal technique is able to find anomaly sources, it is not able

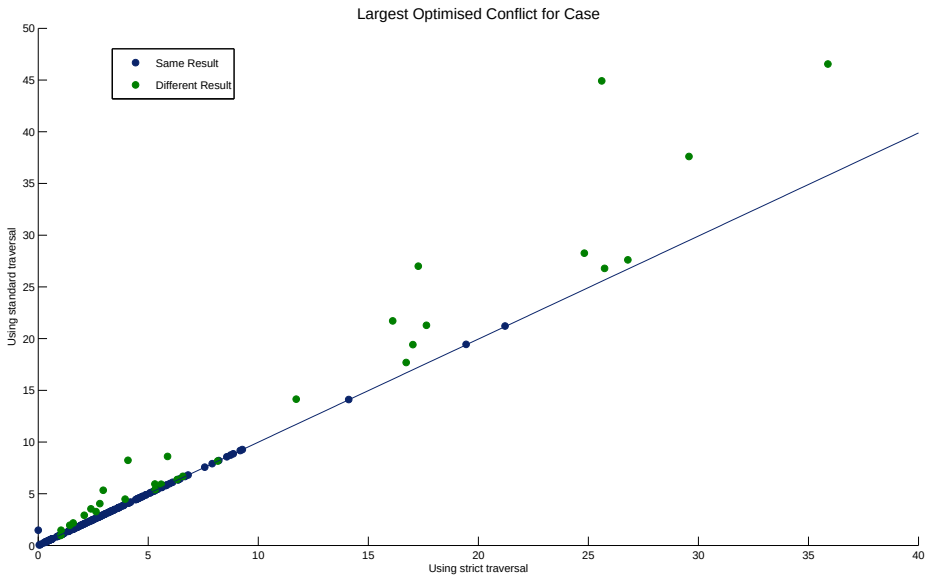

Figure 5: *Effect of traversal technique on largest optimised conflict found*

to report the entire set of observations responsible for the conflict to a human operator. This effect will be amplified in the case of complex conflicts (where the second best branch is likely to be important), where it is most important that the algorithm be able to report the full source of the anomaly to a human operator for interpretation.

In all our tests, the use of the strict traversal technique saved less than ten seconds in searching 200 cases, an efficiency saving which we decided did not justify the loss of performance in other key areas. However, we do note that large, complex networks may be able to benefit from a hybrid approach, exploiting the heteroskedasticity noted above so as to only call the standard traversal on cases which have passed a certain score threshold whilst using strict traversal. Finally, we also tested an exhaustive traversal technique, which searched over every clique at every stage. However, this technique not only suffered from severe computational inefficiency (taking over one hundred times as long as the standard technique), it also returned significantly less concise and intuitive explanations of conflict, whilst providing only a small gain in detection performance.

5 Tests of Anomaly Detection

In order to test the performance of the CLA, we used simulated data from an existing BN built for Maritime Anomaly Detection (MAD). This network was learned from real world data collected from Automatic Identification System (AIS) transmissions from vessels in and around Sydney harbour, following the procedure in [13, 14], and was built using software provided by Bayesian Intelligence Pty Ltd, under contract from DSTO. The AIS system publicly reports both dynamic (e.g. latitude, longitude, speed) and static (e.g. ship type, nationality, size) information in a standard format set by the International Convention for the Safety of Life at Sea. Before learning the model, the data was discretized using the *Snob* software package [19], which discretised continuous data using Minimum Message Length scored finite mixture models. The structure and conditional probability distributions were then learned using the Causal discovery via Mixed Message Length (CaMML) software [23]. The resulting model contained 52 nodes, 94 edges, and a total of 11446 conditional probabilities. Various anomalous detectors were used including: the self-information (i.e. joint probability) of the entire case; the Jensen conflict measure of the entire case; and the largest local maximum of conflict in the case, as reported by the CLA. In order to extend the analysis in section 4 we also tested variations of the standard CLA. However, none of these CLA variations had a statistically significant difference in performance when compared with the standard CLA in any test, and the results are not reported below for the sake of conciseness.

5.1 Techniques for Evaluation of Classifier Performance

The CLA, as well as the existing methods for classification detailed above, map from the set of cases into $\mathbb{R}$, assigning a score to each case. In order to turn these scores into a decision result of *anomaly* or *not anomaly*, we set a threshold T and rule that all cases of with a score greater than T are anomalous, and the inverse. Classification performance was measured using the Receiver Operator Characteristic (ROC), which is independent of T, and by the F_1 score, the Matthews Correlation Coefficient [15], and Youden's J statistic [24], each of which depends on T.

5.1.1 Receiver Operating Characteristic

The ROC of a classifier is a graphical representation of the tradeoff made by lowering T. Specifically, the y-axis shows the proportion of actual anomalous cases which are detected (also known as *sensitivity*) given the proportion of non-anomalous cases which are registered as anomalous given T. The self-explanatory Area Underneath the Curve (AUC) is a prevalent metric in the machine learning community for measuring the overall performance of a classifier, independent of T. A 'blind classifier' which simply knows the proportion of cases which are anomalous and classifies at random will have an AUC of 0.5, whereas a perfect classifier which detects every anomalous case before detecting a single non-anomalous case will have an AUC of 1. Selecting T as the threshold value associated with the point on the curve closest to the point $(0, 1)$ in the ROC space is common practice, and it is this value of T we will use for our remaining metrics. All ROC curves used in our research were computed using [4].

5.1.2 Confusion Matrix

At any given value for T, the performance of a binary classifier can be represented as a *confusion matrix* of the form shown in table 1. The acronyms *TP, FP, FN,* and *TN* stand for True Positive, False Positive, False Negative, and True Negative, respectively. By showing all these values, the confusion matrix provides us with an instant overview of the performance of the classifier at a given T.

Table 1: *Confusion matrix for a binary classifier*

		Reported Class	
		Anomalous	Non-Anomalous
Actual Class	Anomalous	*TP*	*FN*
	Non-Anomalous	*FP*	*TN*

5.1.3 F_1 Score

From the confusion matrix we can easily calculate the *precision* (the proportion of cases which the classifier considers anomalous that are actually anomalous) and the *recall* (the proportion of actually anomalous cases which are detected by the classifier) of the chosen classifier at a given T, given by

$$\frac{TP}{(TP+FP)} \quad \text{and} \quad \frac{TP}{(TP+FN)}$$

respectively. The F_1 score is then given by the harmonic mean of these two values, and is a widely used metric for measuring the ability of a classifier to detect anomalous cases. Note that unlike the metrics below, the F_1 score does not take the true negative rate of the classifier into account. The F_1 score can be calculated directly from the confusion matrix as

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{3}$$

5.1.4 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC)[15], sometimes known as the *phi coefficient* is another canonical performance metric for binary classifications, and essentially measures the correlation between the actual classes and those produced by the classifier. As we would expect from a correlation, a score of $+1$ means we have a perfect classifier, a score of 0 is equivalent to classifying at random, and a score of -1 means we have a perfectly wrong classifier (which, of course, could simply be inverted to give a perfect classifier). The MCC is calculated from the confusion matrix as

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP).(TN+FN)}}. \tag{4}$$

5.1.5 Youden's J Statistic

Youden's J statistic [24] is equivalent to the partial AUC to the left of T, and is calculated from the confusion matrix as

$$J = \frac{TP \cdot TN - FP \cdot FN}{(TP + FN)(FP + TN)}. \tag{5}$$

Youden notes that the J statistic has a binomial distribution with variance

$$\sigma_J^2 = \frac{TP \cdot FN}{(TP + FN)^3} + \frac{FP \cdot TN}{(FP + TN)^3}, \tag{6}$$

and that given the asymptotically normal behaviour of the binomial distributution, J statistics for different classifiers may be compared using a standard t-test, provided we have a sufficient (> 20) number of cases.

5.2 Multiple Sources of Anomaly

In order to test detection in the scenario where cases had multiple sources of anomaly, we simulated 200 cases from the existing BN, 40 of which were anomalous. Anomalous cases were generated by selecting eight variables at random (the same eight for all cases) and switching their values with the values of another case in the simulated dataset. The ROC curves for the different classifiers are shown in figure 6 and their various performance metrics are shown in table 2.

Table 2: *Performance metrics for classifying multiple sources of anomaly*

Classifier	Time (s)	AUC	F_1	MCC	J Statistic
Joint Probability	34	0.92±0.030	0.72	0.69	0.62
Total Jensen Conflict	73	0.95±0.023	0.86	0.83	0.79
CLA	187	0.94±0.027	0.80	0.76	0.69

As we can see, where we have multiple anomalies the classifiers perform close to equally well. In fact, we did not find a statistically significant difference between any pair of classifiers for any of the metrics used, though all three classifiers performed well. Thus, we fail to reject the null hypothesis that these whole-network techniques and the CLA are equally well suited to anomaly detection in the case where the number of anomalous findings is large relative to the number of nodes in the network.

5.3 Single Source of Anomaly

In order to test the ability of the CLA as well as existing measures in finding a single anomalous entry in a case, we used the same simulated set of 200 cases as above. We generated ten anomalies by switching the observations of the `ShipType` node with other cases in the simulated dataset. Again, we used the self-information and Jensen conflict measure as classifiers, as well as the largest conflict reported by the CLA. The ROC curves

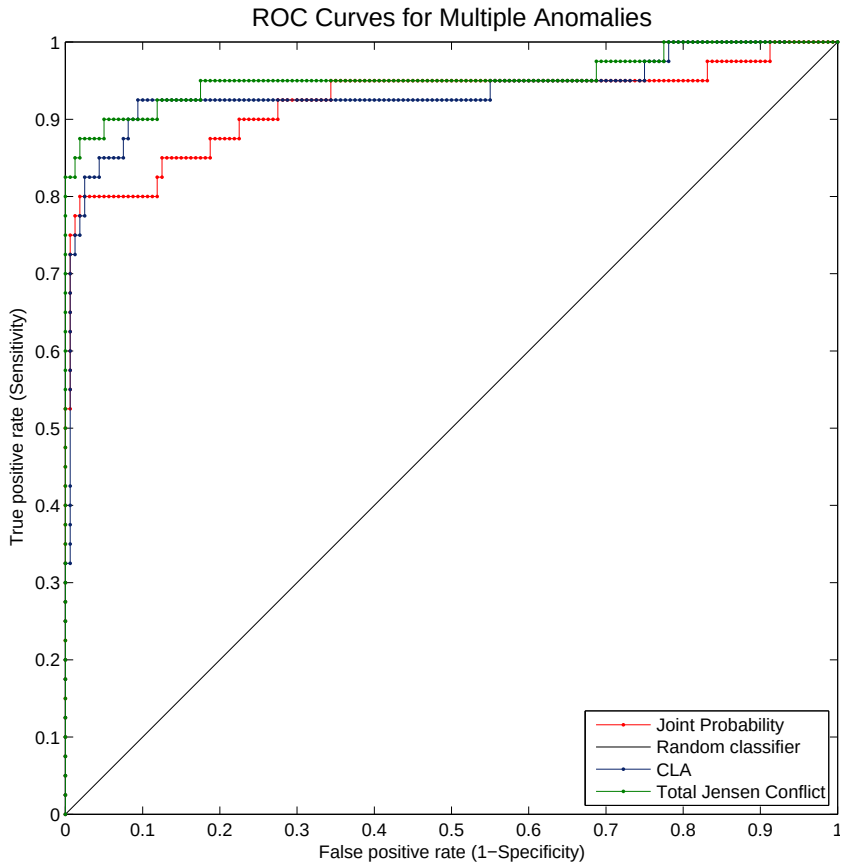

Figure 6: *ROC curves for multiple sources of anomaly*

Table 3: *Performance metrics for classifying a single source of anomaly*

Classifier	Time (s)	AUC	F_1	MCC	J Statistic
Joint Probability	38	0.69±0.095	0.23	0.15	0.07
Total Jensen Conflict	82	0.68±0.095	0.20	0.19	0.10
CLA	193	0.89±0.069	0.29	0.33	0.16

from the classifiers are shown in figure 7 and their performance metrics are shown in table 3.

Here we see the benefit of the CLA. Not only does the CLA significantly outperform the other classifiers in all performance metrics, it also suffers from minimal loss of performance when compared to the multiple anomaly case. A hypothesis test using the Student's t-distribution shows that, compared with the joint probability and Jensen conflict classifiers, the CLA here performs significantly better in AUC, with 1.67σ and 1.72σ distances respectively. Importantly, when compared to their performance on the dataset with multiple anomaly sources, the joint probability and Jensen conflict classifiers experienced a standardised loss of 2.34σ and 2.76σ in the AUCs respectively, both highly significant changes in performance, whereas the performance of the CLA dropped only 0.667σ. We expect this result to be even more pronounced in larger networks.

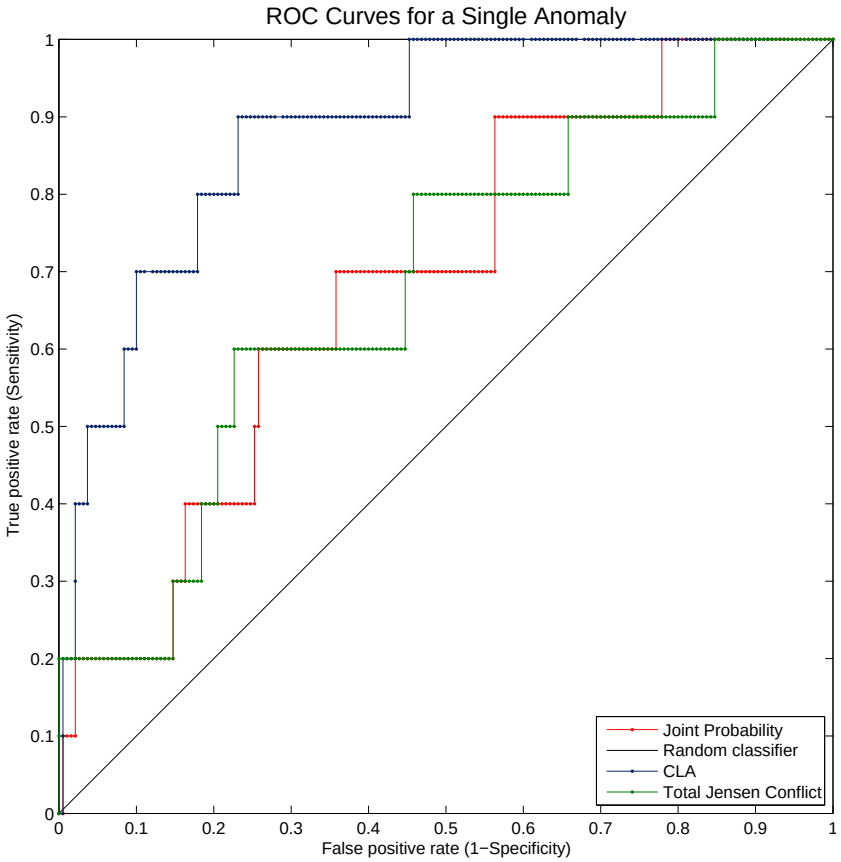

Figure 7: *ROC curves for a single source of anomaly*

6 Tests of Anomaly Attribution

The data above supports our hypothesis that the CLA provides a better approach to anomaly detection against existing methods. However, the true power of the CLA is its capacity for explanation. The ability of the CLA to accurately determine the variables in a case responsible for its anomaly status as well as the most relevant non-anomalous evidence is crucial for humans investigating these anomalies, and is unique to the CLA.

6.1 Single Source of Anomaly

Using the data described in section 5.3 (with `ShipType` as the only anomalous node), and with T set automatically as the point of the ROC curve closest to (0,1), the CLA detected nine of ten anomalies. In eight of these nine detected anomalies, the highest scoring optimised conflict set identified by the CLA optimiser contained the `ShipType` node. In all but one of these cases, the explanation stage of the CLA correctly identified the subset of the conflict set containing `ShipType` as the source of the anomaly. In five of these cases, `ShipType` was identified as the sole anomaly, whilst in the other two cases it was identified in conjunction with other variables. However, even when the actual anomaly was incorrectly reported as being anomalous in conjunction with other observations, the individual

conflict score of each node against the remainder of the network showed `ShipType` to be the most anomalous single node.

Figure 8 shows the explanation generated by the CLA for the highest scoring optimised conflict for one of the anomalous cases in this network. The case was originally a tanker, but we changed its type to 'tug' when generating the anomaly. As we can see, the CLA has identified a conflict set of the `ShipType` node, as well as three neighbouring nodes, `ShipSize, Flag`, and `LongestCloseMyAvgLat`, the average latitude of the ship in question during its longest close interaction with another observed track (set to zero here, indicating no such interactions). The CLA has identified that the values of these three nodes are consistent with each other (with a conflict of -0.85 between them), and that the value of 'tug' for `ShipType` conflicts with this set. Certainly, it makes intuitive sense to us that a tug boat in Sydney harbour which is large, flying a Singaporean flag, and not interacting with any other ships should rightly be classified as an anomaly, and we see a large conflict between these two sets. Lastly, the CLA identifies that the set of three observations is consistent with the rest of the network, whereas the set containing only `ShipType` is in conflict with the rest of the network, and reports that this is the most likely source of the anomaly. Thus in this case (and others not shown here), the CLA has not only managed to determine that this case is anomalous, it has managed to perfectly detect the individual node responsible, and report that information back to a human operator.

```
*** Conflict number 1 with a score of 7.64393106573 ***

Conflicting set contains:
[u'shipSize', u'LongestCloseMyAvgLat', u'flag', u'shipType']

Conflict between set A and set B =        8.49718965961

  Set A:
    Internal Conflict = 1.55053315735e-07
    Conflict With Rest of Net = 3.81611709002
    Contains:
      shipType = Tug     (3.81611709002)

  Set B:
    Internal Conflict = -0.853258748941
    Conflict With Rest of Net = -1.03053566478
    Contains:
      shipSize = v__0_223358605966     (-0.125787481649)
      flag = Singapore     (-0.508587631371)
      LongestCloseMyAvgLat = v_0_0     (-1.28679517942)

SET A IS THE MOST LIKELY SOURCE OF THE ANOMALY
```

Figure 8: *Example of an anomaly explanation*

6.2 Multiple Sources of Anomaly

Quantifying the performance of the explanation stage of the CLA here is far less straightforward than it is in the situation where we have a single anomaly, as we now have multiple sources of anomaly spread across the network, some of which consist of more than one anomalous node. We ran the explanation algorithm on all 40 anomalous tracks in the dataset described in section 5.2, and looked at the conflict set with the greatest conflict score for each. In 36 of these 40 cases, the subset identified by the algorithm as the source

```
*** Conflict number 1 with a score of 18.2920065765 ***

Conflicting set contains:
[u'straightSections', u'main2StopLon', u'Class', u'mainStopLon', u'stops', u'spe
ed5_10Pc', u'courseChangeRate5_10Pc', u'speedSd', u'straightPc', u'main2Course',
 u'courseChangeRate1_5Pc', u'speed1_5Pc']

Conflict between set A and set B =        24.5355871509

  Set A:
    Internal Conflict = 1.74933693697
    Conflict With Rest of Net = 10.6747956802
    Contains:
      stops = v_0_540636042403    (8.82470722474)
      straightPc = v_0_061091790163    (3.55970165405)

  Set B:
    Internal Conflict = -7.99291751143
    Conflict With Rest of Net = 0.00049746726961
    Contains:
      Class = v_6_0    (2.65145210407)
      speed1_5Pc = v_0_173075512529    (1.03226484537)
      straightSections = v_139_007481297    (0.105128229606)
      courseChangeRate1_5Pc = v_0_103857200951    (-0.418073719547)
      speed5_10Pc = v_0_360908751154    (-0.838921649361)
      mainStopLon = v_151_21004377    (-0.937259838232)
      speedSd = v_3_61516809014    (-0.974328848405)
      courseChangeRate5_10Pc = v_0_0304794121126    (-1.17413501112)
      main2StopLon = v_151_210303965    (-1.48110436694)
      main2Course = v_10_4366863152    (-2.02736205159)

SET A IS THE MOST LIKELY SOURCE OF THE ANOMALY
```

Figure 9: *Example of a complex anomaly explanation*

of the anomaly contained at least one actually anomalous node, with an average of 1.83 nodes in the attributed subset over these 36 cases. Furthermore in 21 of these cases, this subset contained no wrongly identified non-anomalous nodes. However, when we used the additional metric of individually checking the conflict of individual nodes in the conflict set against the remainder of the network, the highest scoring node was an anomalous node in all 36 cases.

Figure 9 shows an example of an anomaly explanation from a randomly selected case in this dataset. Here, `stops` and `straightPc` are anomalous nodes, along with six other nodes not included in this conflict set. We see that the explanation algorithm has accurately partitioned the conflict set into the anomalous and non-anomalous nodes, and has correctly labeled set A as the source of the anomaly. Note that both subsets have low internal conflict, indicating that the anomalous nodes agree with each other, which is what we would expect in the case of an actual anomaly, rather than simply garbled data. The simulated anomaly was generated by swapping fields with a different entry in the data set, so we would still expect this kind of internal consistency.

Figure 9 is a good example of a more complex anomaly explanation, though we note that it does not include all anomalous nodes. As explained in section 3.2, the CLA uses the structure of the network learned from non-anomalous cases, and can thus miss relationships between anomalous nodes which did not exist when those nodes where non-anomalous. It is important to note that when this structure exists (which it does here, as all of the anomalies for a given case are generated together), the practice of scoring the elements of the conflict set against the remainder of the network is drawn into question. The demonstrated good performance of the explanation stage of the CLA in this dataset indicates that this is only a concern where the number of anomalous nodes is very large compared with the network.

7 Conclusion

In this paper we have provided an overview of existing techniques in anomaly detection and BNs, and introduced the motivation, underlying logic, and formal specification of the CLA. We have explained the design decisions in creating the CLA, though we acknowledge that further opportunities exist to improve its efficiency, an important consideration when dealing with extremely large datasets. We have presented evidence supporting the performance of the CLA as a classifier, showing it to be a significant improvement over the existing techniques of joint probability and total conflict. Finally, we have demonstrated and tested the unique ability of the CLA to determine the source of an anomaly, and to report it back to a human operator.

We anticipate that the explanation stage of the CLA will be the primary focus of future work on the topic. Generalisations to partitioning multiple differing sources of conflict are a possibility, as well as the use of non-anomalous findings in order to report 'expected values' for these anomalous nodes, providing human operators with even more information into the nature of an anomaly. Details of these techniques, also known as *abductive inference* are given in [22, 18, 5]. We believe that the explanatory power of the CLA provides a powerful and valuable tool for anomaly attribution, and look forward to future work in the area.

References

1. Sakshi Babbar and Sanjay Chawla. On Bayesian network and outlier detection. In *COMAD*, page 125, 2010.

2. Sakshi Babbar, Didi Surian, and Sanjay Chawla. A causal approach for mining interesting anomalies. In *Advances in Artificial Intelligence*, pages 226–232. Springer, 2013.

3. Antonio Cansado and Alvaro Soto. Unsupervised anomaly detection in large databases using Bayesian networks. *Applied Artificial Intelligence*, 22(4):309–330, 2008.

4. Giuseppe Cardillo. ROC curve: compute a Receiver Operating Characteristics curve, 2008. http://www.mathworks.com/matlabcentral/fileexchange/19950, accessed Jan 2014.

5. Urszula Chajewska and Joseph Y Halpern. Defining explanation in probabilistic systems. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 62–71. Morgan Kaufmann Publishers Inc., 1997.

6. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: a survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

7. Norsys Software Corporation. Netica 4.16. Accessed Nov 2013.

8. Eric Driver and Darryl Morrell. Implementation of continuous Bayesian networks using sums of weighted Gaussians. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 134–140. Morgan Kaufmann Publishers Inc., 1995.

9. Andrew Haldane. Why banks failed the stress test. *BIS Review*, 18:2009, 2009.

10. Finn Verner Jensen, Bo Chamberlain, Torsten Nordahl, and Frank Jensen. Analysis in HUGIN of data conflict. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 519–528. Elsevier Science Inc., 1990.

11. Finn Verner Jensen and Thomas Dyhre Nielsen. *Bayesian networks and decision graphs*. Springer, 2007.

12. Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B (Methodological)*, pages 157–224, 1988.

13. Steven Mascaro, Kevin B Korb, and Ann E Nicholson. Learning abnormal vessel behaviour from AIS data with Bayesian networks at two time scales. *Tracks: A Journal Of Artists' Writings*, 2010.

14. Steven Mascaro, Ann Nicholson, and Kevin Korb. Anomaly detection in vessel tracks using Bayesian networks. *International Journal of Approximate Reasoning*, 2013. Submitted.

15. Brian W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

16. Mr. Bayes and Mr. Price. An essay towards solving a problem in the doctrine of chances by the late Rev. Mr. Bayes, FRS, communicated by Mr. Price in a letter to John Canton, AMFRS. *Philosophical Transactions (1683-1775)*, pages 370–418, 1763.

17. Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, Upper Saddle River, 2004.

18. Ulf Nielsen, Jean-Philippe Pellet, and André Elisseeff. Explanation trees for causal Bayesian networks. *arXiv preprint arXiv:1206.3276*, 2012.

19. Jon D Patrick. *Snob: A program for discriminating between classes*. Monash University, Department of Computer Science, 1991.

20. Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *AAAI*, pages 133–136, 1982.

21. Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. *Proceedings of Cognitive Science Society*, CSS-7, 1985.

22. Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

23. Chris Wallace, Kevin B Korb, and Honghua Dai. Causal discovery via MML. In *ICML*, volume 96, pages 516–524. Citeseer, 1996.

24. W.J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. CAVEAT/PRIVACY MARKING

2. TITLE	3. SECURITY CLASSIFICATION
Anomaly Detection and Attribution Using Bayesian Networks	Document (U) Title (U) Abstract (U)
4. AUTHORS	5. CORPORATE AUTHOR
Andrew Kirk, Jonathan Legg and Edwin El-Mahassni	Defence Science and Technology Organisation PO Box 1500 Edinburgh, South Australia 5111, Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO–TR–2975	015-964	Technical Report	July, 2014

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. No. OF PAGES	12. No. OF REFS
2014/1047781/1	AIR 07/324	CDRSRG	22	24

13. DSTO PUBLICATIONS REPOSITORY	14. RELEASE AUTHORITY
`http://dspace.dsto.defence.gov.au/dspace/`	Chief, National Security and Intelligence, Surveillance & Reconnaissance Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DSTO RESEARCH LIBRARY THESAURUS

Bayesian networks, machine learning, situation assessment, situational awareness, maritime surveillance

19. ABSTRACT

We present a novel approach to anomaly detection in Bayesian networks, enabling both the detection and explanation of anomalous cases in a dataset. By exploiting the structure of a Bayesian network, our algorithm is able to efficiently search for local maxima of data conflict between closely related variables. Benchmark tests using data simulated from complex Bayesian networks show that our approach provides a significant improvement over techniques that search for anomalies using the entire network, rather than its subsets. We conclude with demonstrations of the unique explanatory power of our approach in determining the observation(s) responsible for an anomaly.